Choose $e'$ as in (7), then

$$\sum_{i=0}^{\log n} p_{s,i} \leq \sum_{i=0}^{\log n} ((1 - q(s - 1))^{2^i} \cdot 2^i q(s)$$

$$\leq \sum_{i=0}^{\log n} ((1 - q(s - 1))^{2^i} \cdot 2^i c_{\min} q(s - 1)$$

which implies Claim 6. \qed
This can be shown by induction on $N$. Hence,
\[
E[COM_N(H, s)] \geq \prod_{i=1}^{\lfloor S_H \rfloor} (1 - q_i(s - 1)) \sum_{i=1}^{\lfloor S_H \rfloor} q_i(s) \\
> \prod_{i=1}^{\lfloor S_H \rfloor} (1 - q_i(s - 1))c \sum_{i=1}^{\lfloor S_H \rfloor} q_i(s - 1) \\
\geq (1 - \sum_{i=1}^{\lfloor S_H \rfloor} q_i(s - 1)) \cdot cq(s - 1) \cdot |S_H| \\
= (1 - q(s - 1)|S_H|) \cdot cq(s - 1) \cdot |S_H|.
\]

Assume that $q(s - 1) = \frac{1}{2^k}$ for $k > 1$ (if $k \leq 1$ then $q(s) \leq 2q(s - 1)$, therefore Claim 5 holds), and let $|S_H| = 2^{k-1}$, then
\[
E[COM_N(H, s)] > (1 - \frac{1}{2^k} \cdot 2^{k-1}) \cdot c \cdot \frac{1}{2^k} \cdot 2^{k-1} = \frac{c}{4}
\]
which contradicts inequality (5).

\[\square\]

**Proof of Claim 6 for non-uniform algorithms:** We note, that for any $0 \leq i \leq \log n$ and $s > 1$
\[
p_{s,i} \leq \prod_{i=1}^{2^l} (1 - q_i(s - 1)) \sum_{i=1}^{2^l} q_i(s),
\]
then
\[
\sum_{i=0}^{\log n} p_{s,i} \leq \sum_{i=0}^{\log n} \prod_{i=1}^{2^l} (1 - q_i(s - 1)) \sum_{i=1}^{2^l} q_i(s).
\]
Define the function
\[
f(x) = \ln(1 - x).
\]
Since $f$ is concave, Jansen’s inequality \[1\] implies
\[
\sum_{i=1}^{2^l} f(q_i(s - 1)) \leq 2^l f(q(s - 1)).
\]
Exponentiating in both sides we have
\[
\prod_{i=1}^{2^l} (1 - q_i(s - 1)) \leq ((1 - q(s - 1))^{2^l}.
\]
References


Appendix

**Proof of Claim 5 for non-uniform algorithms:** Take $c = 4\delta$, where $\delta$ is defined in (5), and assume that there exists $s > 1$, such that $q(s) > cq(s - 1)$. Note, that for any $N > 1$ and $p_1, \ldots, p_N$, $0 \leq p_i \leq 1$,

$$\prod_{i=1}^{N}(1 - p_i) \geq 1 - \sum_{i=1}^{N} p_i. \quad (13)$$
Hence, we can summarize writing

\[
E[COM_N(j)] \leq \frac{n}{2^k} \sum_{t=1}^{\log n} \prod_{k=1}^{t-1} \left(1 - \frac{2^t}{n}\right)^{n/2^k} \\
\cdot \left(\frac{2q_L^c}{(1 - q_L^c)(1 - 2q_L^c)} \cdot \frac{2^{t+2d}}{n} + \frac{(2q_L^c)^d}{2} \frac{2}{(1 - q_L^c)(1 - 2q_L^c)}\right) \\
\leq \frac{n}{2^k} \sum_{t=1}^{\log n} \prod_{k=1}^{t-1} \left(1 - \frac{2^t}{n}\right)^{n/2^k} \cdot \frac{2^{t+2d}}{n} \cdot O(1).
\]

Following the steps of the proof of the message complexity of the CUP for the reliable case (Theorem 1) we have the statement of the theorem. \(\square\)

We now show that the expected number of cells retransmitted by the sender during algorithm CUP is linear in the maximal number of missing cells.

**Theorem 9** Using the algorithm CUP in a network with sender to receiver reliability parameter \(q_L^c\), for any \(1 \leq r \leq m\), the sender retransmits during the algorithm \(O(r)\) cells with probability \(1 - O(1/n)\).

**Proof:** We note, that for any node in \(i \in S_j\), after the sender retransmits \(2d\) times \(2^{j+1}\) cells in response to NACKS sent from \(S_j\), i has obtained its missing cells with probability \(1 - O(n^{-3})\). Therefore, for any \(1 \leq j \leq \lfloor \log r \rfloor\), the probability that there exists a node \(i \in S_j\) with \(m_i > 0\) is bounded by

\[
1 - (1 - (2q_L^c)^d)^{|S_j|} \leq 1 - O(exp(-1/n)) = O(1/n) .
\]

Thus, with probability \(1 - O(1/n)\) the sender retransmits at most \(2d \sum_{j=0}^{|\log r|} 2^{j+1} = d \cdot 16r\) cells. \(\square\)

### 6 Discussion

A new scheme for reliable multicast using IDA was presented as well as randomized algorithms minimizing the number of rounds and the amount of receiver-to-sender traffic. These algorithms were shown to be optimal, by proving lower bounds on the number of negative acknowledgments and time.

Our paper leaves open several interesting avenues for further research. It is interesting to investigate other topologies, in particular those which allow us to model the dependencies between cell losses for neighboring nodes. Can we take into account the fact that cells are typically lost in batches? How can information about transmission of previous packets be used to tune the redundancy factor? On the more practical side, much work is left to do on designing the protocols which exploit IDA inside an implementation of reliable multicast, defining the exact protocols for session management, flow control and transmission.
Proof: For deriving the bounds we may assume equal loss probabilities for all the receivers (The set of loss probabilities \( q^1_L(1), \ldots, q^N_L(n) \), and \( q^N_L(1), \ldots, q^N_L(n) \), can be replaced by \( \hat{q}_L(i) = \max_{1 \leq i \leq n} q^N_L(i) \) and \( \hat{q}_L^N(i) = \max_{1 \leq i \leq n} q^N_L(i) \forall 1 \leq i \leq n \).

For time complexity recall, that after \( \log n \) phases, each receiver node which has not obtained its missing cells, sends a NACK with probability 1. For computing the bound we assume that \( i \) sends a NACK after \( \log n \) phases, and needs to wait until it receives \( m_i \) cells from the sender. Note, that for each NACK message sent by \( i \), the expected number of phases until a NACK is received by the sender is \( \frac{n}{1 - q^N_L} \). In response to NACK(\( \text{subset}(i), m_i \)) the sender retransmits at least \( 2m_i \) cells in the next phase. Denote by \( Y \) the random variable which counts the number of cells lost in a retransmission of \( 2m_i \) cells to \( i \), then from Markov’s inequality:

\[
\text{Prob}(Y > m_i) < 2q^N_L.
\]

We note, that since (12) holds for any \( 1 \leq m_i \leq m \), if exactly \( i \) NACK messages sent by \( i \) were received by the sender, the probability that \( i \) needs to send another NACK is bounded by \( (2q^N_L) \). Therefore the expected number of phases until \( i \) receives \( m_i \) cells, after it sends the first NACK message, is bounded by

\[
\frac{1}{1 - q^N_L} \sum_{i \geq 1} (2q^N_L)^{-1} = \frac{1}{(1 - q^N_L)(1 - 2q^N_L)} = O(1)
\]

and the overall time complexity is \( O(\log n) \).

For message complexity, it suffices to show that for any \( 1 \leq j \leq H \), the expected number of NACK messages sent by \( S_j \) is bounded by a constant. We write below the total number of messages sent from \( S_j \) as the sum of:

- Messages sent from \( S_j \) until the \( d \)th phase in which a NACK of \( S_j \) was received by the sender. We call each of these phases a successful phase.

- NACKs that were sent from \( S_j \) after its \( d \)th successful phase.

Assume that \( |S_j| = \frac{n}{2^k} \), for some \( 0 \leq k \leq \log n \). Recall, that when receiving the first \( 2d \) NACKs from \( S_j \), the sender retransmits \( 2^{j+1} \) cells, therefore, given that the first NACK was sent from \( S_j \) in phase \( s \), the expected number of NACKs sent from \( S_j \) till the \( d \)-th successful phase is bounded by

\[
\frac{1}{1 - q^N_L} \sum_{i \geq 1} (2q^N_L)^{-1} \cdot \frac{2^{j+1}}{n} \leq \frac{n}{2^k} \cdot \frac{2q^N_L}{1 - 2q^N_L} \cdot \frac{2^{j+1}}{n}.
\]

The expected number of NACKs sent from \( S_j \) after the \( d \)th successful phase is clearly bounded by

\[
\frac{n}{2^k} (2q^N_L)^d \quad \text{and} \quad \frac{1}{1 - q^N_L} \sum_{i \geq 1} (2q^N_L)^{-1} = \frac{n}{2^k} (2q^N_L)^{d+1} \quad \frac{1}{1 - q^N_L} \frac{1}{1 - 2q^N_L}.
\]
prepare \( R = m \cdot 3/2 \) cells;
\( L := R - m \);
put on the shelf \( cell(0), \ldots, cell(L - 1) \);
Start: send \( m \) cells;
\( \text{pointer} := 0; \)
\( t := 1; \)
for \( j := 1 \) to \( \lg m \)
\( \text{counter}(j) := 0; \)
repeat
\( \text{receive a message}; \)
if NACK(\( j_t, m(t) \)) then
\( \text{if counter}(j_t) < 2d \) then \( m_s := 2^{j_t+1}; \)
else \( m_s := 2m(t); \)
\( \text{counter}(j_t) := \text{counter}(j_t) + 1; \)
if \( m_s \leq L \)
\( \text{send } m_s \text{ cells starting at } cell(\text{pointer}) \)
to all receivers;
\( \text{pointer} := (\text{pointer} + m_s) \mod L; \)
else goto Start
\( t := t + 1; \)
until false;

Figure 4: Algorithm CUP: Pseudocode for the sender in the unreliable case

5 Loss of NACKs and Retransmitted Cells

In this section we analyze algorithm CUP for the case where retransmissions of cells and transmissions of NACK messages are unreliable. Specifically, for each receiver node \( i \) and any cell transmitted by the sender, the cell arrives to \( i \) with probability \( 1 - q^L_e(i) \), where \( q^L_e(i) = O(n^{-1/d}) \), for some small constant \( d > 1 \). In addition, a NACK message sent from receiver node \( i \) may be lost with probability \( q^N_L(i) \), where \( q^N_L(i) < 1/4 \), for all \( 1 \leq i \leq n \).

The receivers use the algorithm in Figure 2. The sender uses the CUP algorithm with the following change: For each subset \( S_j, 1 \leq j \leq \lg m \) the sender keeps a counter. When a NACK \( (j_t, m(t)) \) is received at time \( t \) from a member in \( S_{j_t} \), the sender checks the counter of \( S_{j_t} \). As long as the counter is lower than \( 2d \), i.e., less than \( 2d \) NACKs were received from \( S_j \), the sender retransmits \( 2^{j_t+1} \) cells, otherwise, it retransmits \( 2m(t) \) cells over the network. The modified protocol for the sender is given in Figure 4.

Theorem 8 For a network with reliability parameters \( q^L_e \) and \( q^N_L \), algorithm CUP terminates within \( O(\log n) \) phases. The message complexity of the algorithm is \( O(\log m) \).
and since
\[ \sum_{s \geq r} p_{s,i} = 1 - \sum_{s=1}^{r-1} p_{s,i} \]
it follows from (9) that
\[ E[T^A] \geq \frac{r}{\log n + 1} \sum_{i=0}^{\log n} (1 - \sum_{s=1}^{r-1} p_{s,i}) \]
\[ \geq \frac{r}{\log n + 1} (\log n + 1 - c'r) . \]
Taking \( r = \frac{\log n + 1}{2c'} \) we have
\[ E[T^A] \geq \frac{1}{2c'} (\log n + 1 - c'(\frac{\log n + 1}{2c'})) \]
\[ = \frac{\log n + 1}{4c'} \]
which completes the proof.  \( \square \)

### 4.2.2 Non-Uniform Algorithms

We now extend the lower bound of Theorem 4 to non-uniform algorithms, where different receiver nodes may follow different algorithms.

**Theorem 7** For any \( r, 1 \leq r \leq m \) and a randomized algorithm \( A \), if \( A \) sends \( O(\log r) \) NACK messages, then
\[ E[T^A] = \Omega(\log n) . \]  \hspace{1cm} (11)

**Proof:** Denote by \( q_i(s) \) the probability that node \( i \) sends NACK in phase \( s \), and by
\[ q_i(s-1) = 1 - (1 - q_i(1)) \cdot (1 - q_i(2)) \cdots (1 - q_i(s-1)) \]
the probability that node \( i \) sends at least one NACK in the first \( (s-1) \) phases. Since the algorithm is non-uniform, these probabilities may now be different for different nodes. Let
\[ q(s) = \frac{1}{|S_H|} \sum_{i \in S_H} q_i(s) \]
be the average success probability of a node in \( S_H \) in phase \( s \), and
\[ q(s-1) = \frac{1}{|S_H|} q_i(s-1) \]
be the average success probability of a node in \( S_H \) in the first \( (s-1) \) phases.

In the appendix we prove that Claims 5 and 6 hold even when \( A \) is non-uniform, which suffices to prove the theorem.  \( \square \)
Let \( q(s-1) = \frac{1}{2^s} \), for some \( k > 1 \). (If \( k \leq 1 \) than \( q(s) \leq 2q(s-1) \) and we are done.) For \( |S_H| = 2^k \) we have:

\[
E[COM_N(H, s)] \geq (1 - \frac{1}{2^k})^{\frac{1}{2^k} \epsilon^2 2^k} > \epsilon .
\]

which contradicts inequality (5).

Given \( s > 1 \) and \( 0 \leq l \leq \log n \) let \( p_{s,i} = \text{Prob}(A_{s,i}) \).

**Claim 6** For any \( s > 1 \), there exists \( c' \geq 1 \), such that \( \sum_{i=0}^{\log n} p_{s,i} < c' \).

**Proof:** For any \( l, 0 \leq i \leq \log n \) and \( s > 1 \), \( p_{s,i} \leq (1 - q(s-1))^{\frac{1}{2^i}} 2^i q(s) \). Let \( c_{min} \) be the minimal \( c \geq 1 \) satisfying (4), and

\[
c' = 2c_{min} .
\]

then

\[
\sum_{i=0}^{\log n} p_{s,i} < \sum_{i=0}^{\log n} (1 - q(s-1))^{\frac{1}{2^i}} 2^i c_{min} q(s-1) \quad \text{(by Claim 5)}
\]

\[
\leq 2c_{min} q(s-1) \sum_{f=0}^{\log n} (1 - q(s-1))^f
\]

\[
= c' q(s-1) \frac{1 - (1 - q(s-1))^{n+1}}{q(s-1)} < c' .
\]

The expected number of phases until \( S_H \) sends a NACK for the first time is given by

\[
E[T^A] = \sum_{i=0}^{\log n} \frac{1}{\log n + 1} \sum_{i \geq 1} p_{s,i} \cdot s .
\]

Let \( r > 1 \) be an integer (to be determined), then

\[
E[T^A] \geq \frac{r}{\log n + 1} \sum_{i=0}^{\log n} \sum_{s=1}^{\frac{r}{2} \log n} p_{s,i} .
\]

In addition, from the definition of \( p_{s,i} \) and from Claim 6

\[
\sum_{i=0}^{\log n} \sum_{s=1}^{\frac{r}{2} \log n} p_{s,i} \leq c' r ,
\]

11
4.2.1 Uniform Algorithms

We start with the simpler case where the algorithm is uniform, i.e., all receiver nodes follow the same algorithm, regardless of their index.

**Theorem 4** Fix \( r, 1 \leq r \leq m \) and let \( A \) be a uniform randomized distributed algorithm with redundancy ratio \( \mu_A > 1 \). If \( A \) sends \( O(\log_{\mu_A} r) \) NACK messages, then

\[
E[T^A] = \Omega(\log n).
\]  

(2)

**Proof:** Let \( H = \lfloor \log_{\mu_A} r \rfloor \). As in the proof of Theorem 3, we partition the receiver nodes into \( H \) subsets, \( S_1, \ldots, S_H \), such that receiver node \( i \) belongs to \( S_j \) if \( \mu_A^{-1} \leq m_i < \mu_A^j \), for some \( j, 1 \leq j < H \). We prove that the expected number of phases until \( S_H \) sends the first NACK message is \( \Omega(\log n) \).

To compute \( E[T^A] \), we average over all possible sizes of \( S_H \). In particular, assume that \( |S_H| = 2^i \), where \( i \) is chosen uniformly in the range \( \{0, \ldots, \log n\} \). Let \( A_{i,s} \) be the event “The first success of \( S_H \) is in phase \( s \)”, where \( |S_H| = 2^i \). We denote by \( q(s) \) be the probability that a node sends NACK in phase \( s \), then

\[
q(s) = 1 - (1 - q(1)) \cdot (1 - q(2)) \cdots (1 - q(s - 1))
\]

is the probability that node \( i \) in \( S_H \) sends at least one NACK in the first \((s - 1)\) phases.

**Claim 5** If

\[
E[COM_N] = O(\log r),
\]  

(3)

then there exists a constant \( c \geq 1 \), such that for any \( s > 1 \),

\[
q(s) < c \cdot q(s - 1).
\]

(4)

**Proof:** For any subset \( S_j, 1 \leq j \leq H \), \( E[COM_N(j)] \geq 1 - \epsilon \). By Equation (3), and since \( H = \lfloor \log_{\mu_A} r \rfloor \), this implies that each subset \( S_j \) sends \( O(1) \) messages during the execution of the algorithm. Therefore, there exists \( \delta > 1 \) such that for any \( j, 1 \leq j \leq H \),

\[
E[COM_N(j)] \leq \delta.
\]  

(5)

Let \( E[COM_N(j,s)] \) be the expected number of NACK messages sent by \( S_j \) in phase \( s \). Observe that for \( S_H \)

\[
E[COM_N(H,s)] \geq (1 - q(s - 1))^{\lceil S_H \rceil q(s)} \cdot |S_H|.
\]  

(6)

Let \( c = \delta \epsilon^2 \), and assume, that there exists \( s > 1 \), such that \( q(s) > cq(s - 1) \), then we show that there exists \( |S_H| \), such that the expected number of NACK messages sent by \( S_H \) is larger than \( \delta \).
The next theorem shows a lower bound of $\Omega(\log m)$ on the number of NACK messages sent by a randomized algorithm $A$ for reliable multicast.

**Theorem 3** Fix $r$, $1 \leq r \leq m$ and $\epsilon$, $0 < \epsilon < 1$, and let $A$ be a randomized distributed algorithm with redundancy ratio $\mu_A > 1$. If $A$ guarantees that all receivers obtain the original packet with probability at least $1 - \epsilon$, then the expected number of NACK messages sent by $A$ satisfies

$$E[COM_N] \geq (1 - \epsilon) \log_{\mu_A} r.$$

**Proof:** Observe, that if $R_A(m(t)) \leq \mu_A \cdot m(t)$, for all $m(t), 1 \leq m(t) \leq m$, then in order to guarantee the retransmission of $r$ cells by the sender, at least one receiver $i$ with $m_i \geq \frac{r}{\mu_A}$ has to send a NACK message with probability higher than $1 - \epsilon$.

We partition the receiver nodes into $H = \lceil \log_{\mu_A} r \rceil$ subsets, $S_1, \ldots, S_H$, such that receiver node $i$ belongs to $S_j$ if $\mu_A^j \leq m_i < \mu_A^j$, for some $j, 1 \leq j < H$.

It suffices to have a single member of $S_H$ send a NACK message. We argue however, that least one NACK message should be sent by some node in $S_j$ with probability higher than $1 - \epsilon$, for any $j, 1 \leq j \leq H$. Otherwise, consider the execution in which $H = j$; since $r$ is unknown and the receiver nodes do not communicate with each other, the nodes in $S_j$ obtain their missing cells with probability smaller than $1 - \epsilon$.

Let $E[COM_N(j)]$ be the expected number of NACK messages sent by nodes in $S_j$. The total number of NACK messages sent by $A$ satisfies

$$E[COM_N] = \sum_{j=1}^{H} E[COM_N(j)]$$

$$\geq \sum_{j=1}^{H} \text{Prob}(\text{a node in } S_j \text{ sends NACK}) \quad \text{(by Markov inequality [5])}$$

$$\geq \sum_{j=1}^{H} (1 - \epsilon)$$

which yields inequality (1). \qed

### 4.2 Time Complexity

We now derive a lower bound on the expected number of phases required for the reliable multicast of a packet. Let $E[T^A]$ denote the expected number of phases until $S_H$ sends a NACK message for the first time. Since NACK transmission and cell retransmission are reliable, this guarantees that the algorithm terminates.
\[
\begin{align*}
&\leq \frac{1}{2^k} + \sum_{j=1}^{\log n - 1} \frac{2^{j-k}}{\exp(2^{j-k-1})} \\
&= \frac{1}{2^k} + \epsilon \sum_{j=1}^{\log n - 1} \frac{2^{j-k}}{\exp(2^{j-k})} \\
&\leq \frac{1}{2^k} \epsilon \sum_{j=1}^{\lfloor \log n \rfloor + 1} 2^j + \sum_{j=1}^{\log n} \frac{\epsilon 2^j}{\exp(2^j)} < 7\epsilon,
\end{align*}
\]

which implies the theorem.

\[\square\]

4 Lower Bounds

We now prove lower bounds for randomized algorithms for reliable multicast in the IDA-based scheme. The lower bounds are proved under the assumption that NACK transmission and cell retransmission are reliable (as in the previous section); however, they match the upper bounds obtained in the next section, where these assumptions are removed.

4.1 Message Complexity

In this section, we derive a lower bound on the number of NACK messages sent by any distributed randomized algorithm for reliable multicast. In the rest of this section, let \( r \) denote the maximal cell deficit after the original transmission; note that \( 0 \leq r \leq m \). Since we are concerned with minimizing the total amount of cells transmitted by the algorithm, we restrict our attention to algorithms which guarantee that the number of cells retransmitted is at most \( \epsilon \) times \( r \), for a small constant \( \epsilon > 1 \).

**Definition 1** For \( t > 1 \), \( 1 \leq m(t) \leq m \), let \( R_A(m(t)) \) denote the number of cells retransmitted by the sender in response to a NACK message \((j, m(t))\) at time \( t \), under algorithm \( A \). Then the redundancy ratio of \( A \) is

\[ \mu_A \equiv \max_{1 \leq m(t) \leq m} \frac{R_A(m(t))}{m(t)}. \]

Clearly, \( \mu_A \geq 1 \); for obtaining the lower bounds we can assume that \( \mu_A > 1 \).

Assume that during the execution of an algorithm \( A \) the sender gets NACK messages at times \( t_1, \ldots, t_k \). If the redundancy ratio of an algorithm is \( \mu_A \), then at time \( t_i \) the sender retransmits at most \( \mu_A \cdot m(t_i) \) cells. Since retransmissions are reliable, \( \sum_{i=1}^{k} m(t_i) = r \). Thus:

**Lemma 2** If the redundancy ratio of \( A \) is \( \mu_A \), then the total number of cells retransmitted under \( A \) satisfies \( E[COM_c] \leq \mu_A \cdot r \).
Figure 3: Algorithm CUP: Pseudocode for the sender.

Let $E[COM_N(j)]$ be the expected number of NACK messages sent by nodes in subset $S_j$, $1 \leq j \leq H$. Since

$$E[COM_N] = \sum_{j=1}^{H} E[COM_N(j)] ,$$

it suffices to show that $E[COM_N(j)]$ is bounded by a small constant, for any $j$, $1 \leq j \leq H$, in order to bound the message complexity of algorithm CUP.

For any $j$, $1 \leq j \leq H$, write $|S_j|$ as $\frac{n}{2^k}$ for some $k$, $0 \leq k \leq \log n$. Let $A_i^j$ be the event “the first round in which a node in $S_j$ sends a NACK message is $s$”. We can write:

$$E[COM_N(j)] = \sum_{s=1}^{\log n} E[COM_N(j) | A_i^j] \cdot \text{Prob}(A_i^j)$$

$$A_i^j = \left[ COM_N(j) | A_i^j \right] \cdot \text{Prob}(A_i^j) + \sum_{s=2}^{\log n} E[COM_N(j) | A_i^j] \cdot \text{Prob}(A_i^j)$$

$$\leq \frac{1}{2^k} + \frac{n}{2^k} \sum_{s=1}^{\log n-1} \frac{2^s}{n} \prod_{t=1}^{s} \left( 1 - \frac{2^{t-1}}{n} \right)^{n/2^k}$$

$$\leq \frac{1}{2^k} + \frac{n}{2^k} \sum_{s=1}^{\log n-1} \frac{2^s}{n} \cdot \exp \left( \sum_{t=0}^{s-1} -2^{t-k} \right) \quad \text{(by a standard approximation)}$$
Given a packet of length $L$, assume that the amount of data that can be transmitted in a single cell is $L/m$, for some $m$. The sender initially generates $R = 3m/2$ pieces from a packet, using the information dispersal algorithm, with $R$ and $m$ as parameters; this implies that a receiver node can reconstruct the original packet from any subset of $m$ cells. The sender initially assumes that the network is reliable and sends exactly $m$ cells; The remaining cells are kept “on the shelf”, in a cyclic list, for retransmissions; this means a storage overhead of 50%.

When the sender receives a NACK($j_t$, $m(t)$) message at time $t > 0$, indicating a deficit of $m(t)$ cells in receiver node that belongs to $S_{j_t}$, it responds as follows.

- If $2^{j_t} > m/2$ then the sender retransmits the $m$ cells originally computed for the packet,
- otherwise, the sender transmits the next $m(t)$ cells on the shelf.

The extra cells are kept only for the duration of the CUP algorithm, i.e., for $O(\log n)$ phases, after which these cells are discarded. The pseudocode appears in Figure 3.

We now prove that algorithm CUP guarantees that each receiver node obtains $m$ cells within $O(\log n)$ phases, and that $O(\log m)$ NACK messages are sent by the algorithm.

To simplify the calculations we assume throughout the paper that $n$ is a power of 2. However, our results hold for any $n \geq 1$; for the case where $2^i < n < 2^{i+1}$ one can choose $n' = 2^i$ and assume that $(n - n')$ receiver nodes have initial deficit $m_i(0) = 0$.

**Theorem 1** Algorithm CUP terminates within $O(\log n)$ phases, and its message complexity is $O(\log m)$.

**Proof:** Clearly, for each receiver node $i$, after $\log n$ phases either $i$ already obtained its missing cells, or $i$ sends a NACK message with probability 1, and terminates. This proves that the algorithm terminates within $O(\log n)$ phases.
Additionally, we consider the storage overhead of the algorithm. When IDA is used for breaking the original packet into \( R \) cells, extra cells are stored to be used by the sender during retransmission phases. The algorithms presented in this paper are space efficient, as they generate at most \( 3/2m \) cells.

3 Reliable Retransmission

In this section, we present a randomized distributed algorithm that guarantees the transmission of at least \( m \) different cells to each receiver. This section considers a simplified version of the problem, in which we assume the initial transmission of cells is unreliable, but all later retransmissions are reliable. Section 5 below addresses the more realistic case where retransmissions are not reliable.

Assume \( m_i \) cells are missing in receiver node \( i \) after the initial transmission, where \( 0 < m_i \leq m \). Since retransmission is reliable it suffices to guarantee that the sender knows \( \max m_i \), so it can retransmit additional cells. Thus, if a receiver node \( i \) sends NACK(\( i, m_i \)) to the sender, then it will get at least \( m_i \) cells in the next phase, satisfying its cell deficit.

Intuitively, the algorithm assumes the set of receivers is partitioned into \( \log m \) subsets \( S_1, \ldots, S_{\log m} \); receiver node \( i \) is in \( S_j \) if \( 2^{j-1} \leq m_i \leq 2^j \). Clearly, each receiver \( i \) knows to which subset \( S_j \) it belongs.

Let \( H = \max_{1 \leq i \leq n} \lfloor \log m_i \rfloor \).\(^2\) Since nodes in \( S_H \) have the highest cell deficit, it suffices that some node in \( S_H \) will send a NACK message. However, \( H \) is unknown, and therefore receivers in every subset \( S_j \) must assume that \( H = j \) and attempt to have a member of \( S_j \) send a NACK message.

Randomization is used by receivers in \( S_j \) to minimize the number of NACKs. At time \( t \), \( t \geq 1 \), a receiver in \( S_j \) tosses a coin and sends a NACK with some probability \( \rho(t) \).

If each receiver node \( i \) knows the size of \( S_i \), where \( l = \lfloor \log m_i \rfloor \), then we can employ an iterative algorithm. In each round, node \( i \) sends a NACK message with fixed probability \( \rho = \frac{1}{|S_i|} \), where \( S_i \) is the subset node \( i \) belongs to. It can be shown that each receiver node obtains \( m \) cells with probability \( 1 - 1/m \) within \( O(\log m) \) rounds and that \( E[COM_N] = O(\log m) \).

Unfortunately, the receivers do not know the size of their subsets. Instead, each node modifies its selection probability, and keeps tossing the coin until it gets “Heads”, or it receives the required number of cells (due to a NACK message sent by other nodes).

The algorithm operates in phases; initially, \( m_i(1) = m_i \). In each phase \( t \geq 1 \), if \( m_i(t) = 0 \) then receiver node \( i \) stops; otherwise, it sends NACK with probability \( \rho := 2^{t-1}n^{-1} \). It waits until it receives \( N_i \) cells, updates the number of missing cells and goes to the next phase. The pseudocode for algorithm \textit{choose with unknown partition (CUP)} appears in Figure 2.

\(^2\)Unless specified otherwise, all the logarithms in this paper are to the base of 2.
2 Preliminaries

We assume a one-to-many communication situation, where a single sender, $P_s$, has to transmit packets to $n$ receivers, $P_1, \ldots, P_n$, over an ATM network. Packets are broken into cells. In the reliable multicast problem we refer to the retransmission of a single packet by the sender to a set of receiver nodes, until all receivers can reconstruct the packet from the cells they obtained.

Any multicast of a cell over the network may result in a loss, meaning that the cell is not received by some of the receivers. Let $q_{L}^c(i)$ denote the loss probability of a cell transmitted to receiver node $i$, $1 \leq i \leq n$. Typically, $q_{L}^c(i) = O(n^{-1/d})$, where $d > 1$ is a small constant. Indeed, this implies that the expected number of lost cells in the initial transmission is small, i.e., $O(mn^{-1/d})$. However, our reliability multicast problem arises already for the case where a small number of cells are missing for some of the receivers. Note, that when $n$ grows larger this probability becomes non negligible, even when $q_{L}^c(i)$ is relatively small. In particular, if $q_{L}^c(i) > bn^{-1/d}$, with $b, d > 1 \forall 1 \leq i \leq n$, then the probability that at least one receiver did not obtain the $m$ cells in the initial transmission of the packet is $1 - O(e^{-\sqrt{n}})$.

We define the sender to receiver reliability parameter of the network as

$$q_{L}^s = \max_{1 \leq i \leq n} q_{L}^c(i).$$

In addition, a NACK message sent from receiver node $i$ may be lost with probability $q_{L}^N(i)$. We denote by

$$q_{L}^N = \max_{1 \leq i \leq n} q_{L}^N(i)$$

the receiver to sender reliability parameter of the network.

Using Rabin’s IDA algorithm [8] with the parameters $R, m > 1$, each packet is broken to $R$ pieces, such that the complete packet can be reconstructed from any subset of $m$ pieces. Thus, each receiver needs to obtain at least $m$ different pieces, but it does not matter which. For any $t \geq 1$, let $m_i(t)$ denote the number of missing cells in receiver node $i$ at time $t$. Let $m_i$ be the initial cell deficit of receiver node $i$; $m_i(0) = m_i$, for any $i, 1 \leq i \leq n$.

The sender retransmits cells over the network in response to NACK messages sent by receivers. We assume communication is synchronous, and each phase is either a retransmission phase (from sender to receivers) or a NACK phase (from receivers to sender). Receivers do not communicate with each other, and receiver node $i$ has no information on the values of $m_j$ for $j \neq i$. Receiver node $i$ terminates at the first time $t$ in which $m_i(t) = 0$; the algorithm terminates when all receiver nodes terminate.

We measure the running time of an algorithm—the time until it terminates, denoted by $T$. We also measure the message complexity—the number of NACK messages sent during the execution of the algorithm, denoted by $COM_N$. Since randomization is employed, the goal is to minimize $E[T]$ and $E[COM_N]$, where averaging is done over the coin flips of the algorithm.
subset of $m$ pieces suffices to reconstruct $M$. These pieces are symmetric and any subset can equally be used. Thus, if two receivers $P_i$ and $P_j$ each miss exactly one cell, $c_i$ and $c_j$, then a single cell $c'$ not already held by either of them suffices to reconstruct $M$, even if $c_i \neq c_j$.

The benefits of an IDA-based scheme are two-fold: First, the same pieces can be used to compensate for the cell deficits of different receivers. Second, the sender can initially send a surplus of pieces, depending on the estimated reliability of the network (the probability of pieces being lost).

Under this scheme, all the sender has to know is the maximal number of cells missing for some receiver; to achieve this, receivers send negative acknowledgments (NACK), with the number of pieces they miss, to the sender. This paper focuses on algorithms for managing this scheme. We are interested in minimizing the running time and message complexity, i.e., the number of NACKs sent during the execution of the algorithm.

Consider, for example, the simple algorithm under which at time $t$, any receiver node $i$ sends a NACK message to the sender, containing the number of cells it is missing. This algorithm is likely to terminate after $O(1)$ phases but it sends $\Omega(n)$ messages. At the other extreme, consider an algorithm in which every receiver node still missing cells sends a NACK message specifying the number of cells it needs, with some fixed probability $p$, $0 < p \leq 1/n$. The algorithm terminates within $\Omega(n)$ phases, but it has $O(\log m)$ message complexity (this follows from a general theorem we prove below, Theorem 1).

Our objective is to develop algorithms that require a minimal number of rounds before each of the receivers has all necessary $m$ cells, and the smallest number of NACK messages.

We present a randomized algorithm which terminates within $O(\log n)$ phases, and sends $O(\log m)$ NACK messages. The algorithm is first analyzed for the case where NACKs and retransmissions are reliable (Section 3) and later, for the case they are not (Section 5). We also prove lower bounds. If an algorithm starts with maximal cell deficit $r$, retransmits at most $\mu r$ cells, and guarantees that all receivers obtain the original packet with probability at least $1 - \epsilon$, for some small $\epsilon > 0$, then the expected number of NACK messages is at least $(1 - \epsilon) \log_p r$. Moreover, if the algorithm is message optimal, then its expected time complexity is $\Omega(\log n)$. (The lower bounds appear in Section 4.)

The basic IDA-based scheme is inspired by the work on TCP Boston [2], where IDA is used to implement reliable point to point communication. TCP Boston also specifies how to integrate signaling and congestion control mechanisms, issues we do not address here. Since TCP Boston considers point-to-point communication, there is no concern for minimizing receiver-to-sender traffic. The multicast situation poses non-trivial algorithmic concerns, which are the focus of our work.

Part of the proof of the lower bound on the expected step complexity draws ideas from the work of Kushilevitz and Mansour [6], which shows a lower bound on the expected time for broadcast in radio networks.
3. The leaf nodes cannot communicate directly to each other with this connection type.

Multicast requires a single source node (sender) to send a packet to a set of receiver nodes. A point-to-multipoint connection can be used to implement multicast on ATM networks. When a packet is longer than 53 bytes, it should be fragmented into cells, which are sent in sequence over the point-to-multipoint connection. Delivery of cells in ATM networks is best-effort, so cells can be lost, typically due to congestion at intermediate switches along the route from the sender to the receivers. When cells comprising a fragmented packet are lost, the packet cannot be reassembled and received. Thus, native packet delivery on ATM point-to-multipoint connections is not reliable.

This paper addresses the issue of implementing reliable multicast, where delivery of packets to all receivers is guaranteed; there are several systems based on reliable multicast, e.g., Isis [3], RMP [10], Transis [4], and XTP [9].

The reliable multicast problem in ATM networks can be captured by the following abstract problem. A single sender node, $P_s$, has to send a packet $M$ to $n$ receiver nodes, $P_1, \ldots, P_n$; the server is connected to all receivers directly. (See Figure 1.) The packet $M$ is too big to be sent in one piece, so the sender must break it into smaller pieces. The network is unreliable, and some pieces may be lost; furthermore, different pieces may be lost on the way to different receivers. To reassemble the packet, a receiver should have all the pieces.

A simple approach to fragmentation is to “drop” the packet into pieces, and send them to the receivers. When some pieces are lost and do not arrive at some receiver node, the sender can either retransmit all pieces to all receivers, or have each receiver notify which pieces it misses, and send these particular pieces. Under the first option, it is possible that many pieces are wasted; the second option requires significant receiver-to-sender traffic.

We suggest a different scheme, based on the information dispersal algorithm (IDA) of Rabin [8].

\footnote{IDA is related to MDS codes [7, Ch. 11].} In this scheme, the sender produces $R > m$ pieces from the packet $M$ such that any...
IDA-Based Protocols for Reliable Multicast*

Hagit Attiya† Hadas Shachnai‡

Department of Computer Science
The Technion, Haifa 32000, Israel

Abstract

We suggest a new scheme for reliable multicast in ATM networks, based on Rabin’s information dispersal algorithm (IDA). Given a packet of size $m$, that has to be transmitted to $n$ receiver nodes, the paper presents a randomized algorithm which terminates within $O(\log n)$ phases, and sends $O(\log m)$ NACK messages, on the average; its storage complexity is proportional to the size of the original packet, while the number of retransmissions is linear in the number of cells lost in the initial transmission of the packet. The average message and time complexities of the algorithm are proved to be within a constant multiplicative factor of the optimum.

1 Introduction

Asynchronous Transfer Mode (ATM) networks provide fast and high-bandwidth communication among nodes which are connected through a collection of switches. ATM networks require nodes to establish a connection between them before transmitting a sequence of 53-byte long cells from one node to another. ATM supports point-to-point, point-to-multipoint and multipoint-to-multipoint connections. As specified by the ATM forum, a point-to-multipoint connection is a collection of links, and associated endpoint nodes, with the following properties:

1. One ATM link, called the root link, serves as the root in a simple tree topology. When the root node, adjacent to the root link, sends information, all of the remaining nodes on the connection, called leaf nodes, receive copies of the information.

2. Each of the leaf nodes on the connection can send information directly to the root node. The root node cannot distinguish which leaf is sending information without additional (higher layer) information.

---

*This work is supported by grant number 854-6195 from the Israeli Ministry of Science.

†hagit@cs.technion.ac.il

‡hadas@cs.technion.ac.il